МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ УЧРЕЖДЕНИЕ

КАЛИНИНГРАДСКОЙ ОБЛАСТИ

ПРОФЕССИОНАЛЬНАЯ ОБРАЗОВАТЕЛЬНАЯ ОРГАНИЗАЦИЯ

«КОЛЛЕДЖ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И СТРОИТЕЛЬСТВА»

(ГБУ КО ПОО «КИТиС»)

|  |  |  |
| --- | --- | --- |
| Курсовой проект  допущен к защите  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (дата)  Зам.директора по УМР  \_\_\_\_\_\_\_\_\_\_\_\_\_Павленко Г.Я.  (подпись) |  | Курсовой проект  защищен с оценкой  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (оценка)  Руководитель работы  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись, дата) |

Пояснительная записка к курсовому проекту

по дисциплине: МДК 01.01 Разработка программных модулей

Тема Разработка приложения «Агрегатор заданий»

Специальность 09.02.07 «Информационные системы и программирование»

|  |  |
| --- | --- |
| Выполнил: | студент 3 курса,  группы ИСп 19-1  Петров Данил Сергеевич  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) |
| Руководитель: | Большакова-Стрекалова Анна Викторовна, преподаватель ГБУ КО ПОО КИТиС  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) |

Калининград

2022

Рецензия

В процессе использования и тестирования приложения было выявлено, что приложение сделано хорошо, ошибок не обнаружено. Приложение выполняет свои функции исправно, весь требуемый функционал присутствует. Приложение сохраняет все записи и выводит их при его перезагрузке.

Были выявлены следующие недостатки приложения:

* отсутствие анимаций;
* дизайн требует доработок;
* требуется добавление нового функционала.

В итоге можно сделать вывод, что приложение имеет большой потенциал и может быть полезным для пользователей, в случае если поддержка проекта не прекратиться.

Ссылка на репозиторий проекта на GitHub – https://github.com/Kotofeu/Organizer

Содержание

|  |  |
| --- | --- |
| Введение | 4 |
| 1 Описание предметной области | 5 |
| 1.1 Аналоги разрабатываемого приложения | 5 |
| 1.2 Техническое задание | 7 |
| 1.3 Описание структуры приложения | 9 |
| 2 Описание разработки приложения | 11 |
| 2.1 Обоснование средств разработки | 11 |
| 2.2 Разработка интерфейса | 17 |
| 2.3 Разработка логики работы приложения. Схема взаимодействия компонентов проекта | 20 |
| 2.4 Описание переменных, компонентов, классов и подпрограмм | 21 |
| 3 Тестирование и установка приложения | 30 |
| Заключение | 35 |
| Список использованной литературы | 36 |
| Приложение. Термины | 37 |
| Приложение А. Листинг рабочего кода | 38 |
| Приложение Б. Руководство пользователя | 53 |

Введение

Назначения данного курсового проекта является разработка мобильного приложения «Агрегатор заданий». Для выполнения задание следует определить актуальность темы, её цели и задачи для её достижения.

Актуальность: данное приложение крайне популярно среди пользователей и имеет большой потенциал в росте аудитории, а также решает большое количество повседневных пользовательских задач.

Цель данной работы: создание мобильного приложения, для записи заметок и списка задач. Приложение должно обладать дружелюбным для пользователя интерфейсом и приятным дизайном. Приложение будет помогать пользователю не забывать важные для него вещи и предоставлять доступ к ним в любое время.

Задачи для достижения цели: выявить ключевые потребности пользователей, определить целевую аудиторию, разработать красивый минималистичный дизайн и простую навигацию. После составления графического интерфейса необходимо разобрать программу на несколько модулей и скомпилировать проект. После выпуска программного продукта, нужно составить набор тестов и воспроизвести их.

1. Описание предметной области
   1. Аналоги разрабатываемого приложения

Для определения функциональных требований пользователя к приложению, следует разобрать несколько наиболее популярных аналогов. Исследовав рынок, было принято решения разобрать 3 приложения: Taskade [1], Roam Research [2] и Notion [3].

Taskade. Предложение для команды, позволяющее организованно работать в команде. Предоставляет функционал совместного списка задач и ментальных карт. Окно работы приложения предоставлено на рисунке 1.

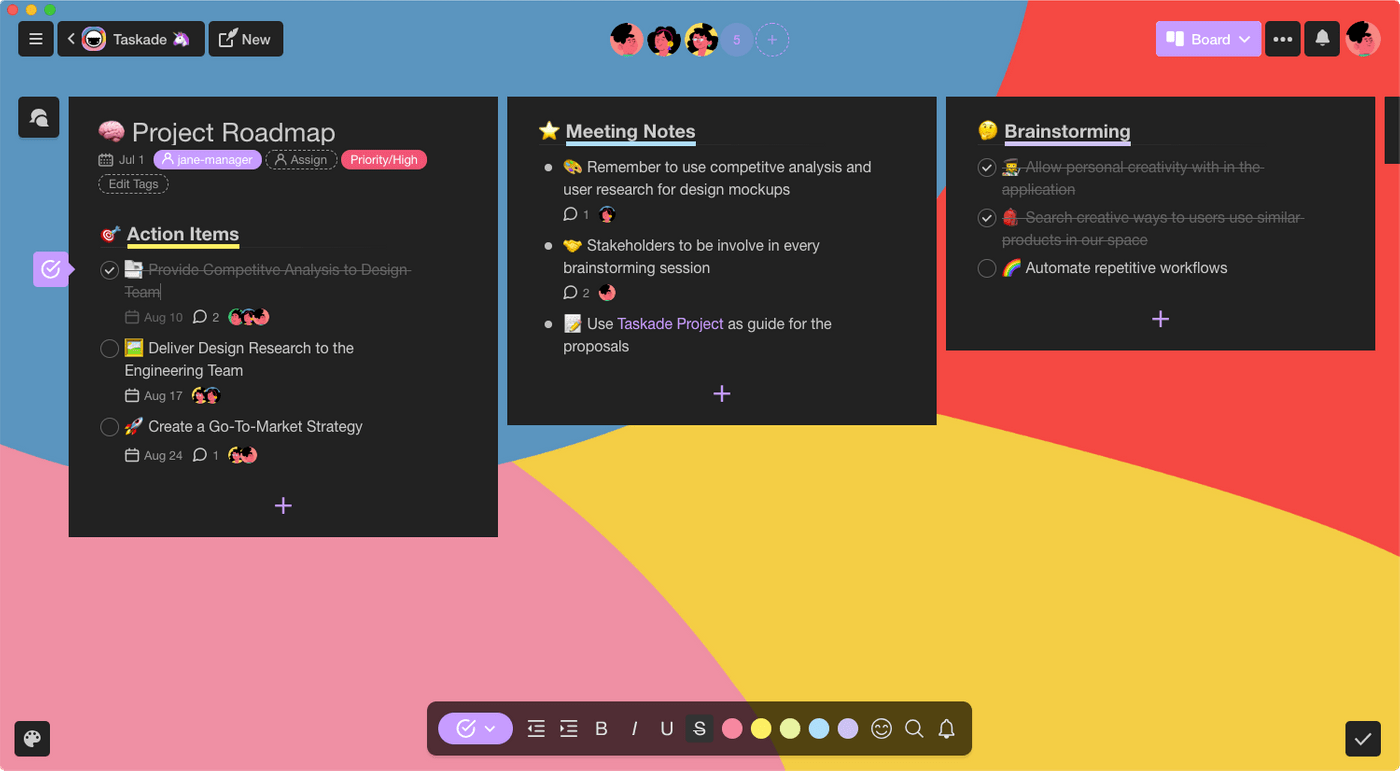


Рисунок 1 – окно приложения Taskade

Минусы решения:

1. Перегруженный и сложный интерфейс продукта.
2. Высокая цена подписки – 5$.
3. Продукт плохо работает на Android устройствах.
4. Отсутствие возможности выставления уведомлений.

Roam Research – веб-приложение для заметок, позволяющая записывать и связывать свои заметки. Окно работы приложения предоставлено на рисунке 2.

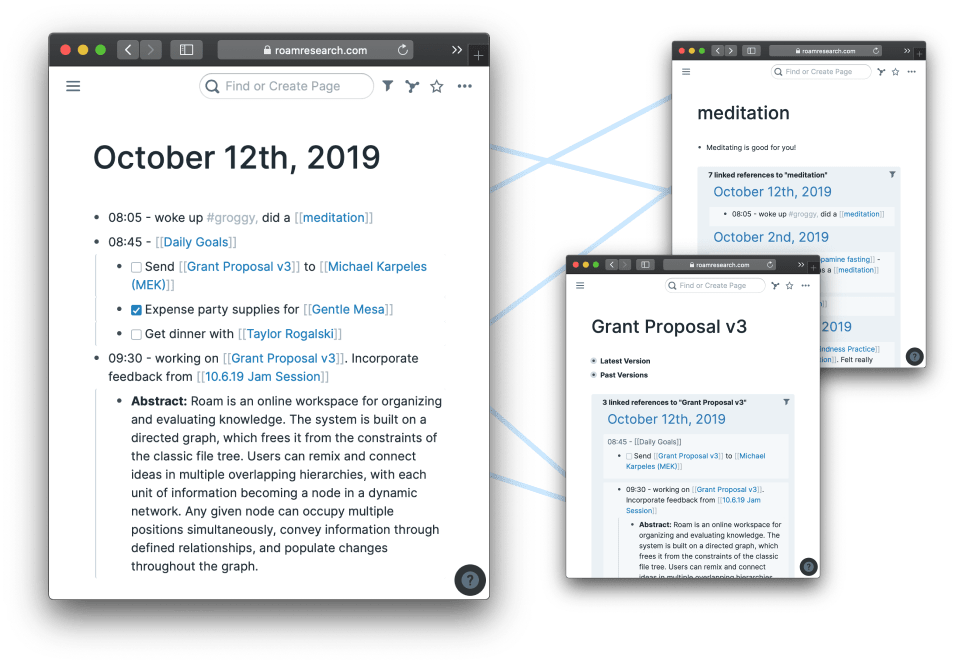


Рисунок 2 – окно работы приложения Roam Research

Минусы решения:

1. Отсутствие мобильного приложения.
2. Сложный импорт существующих заметок.
3. «Плоский» графический интерфейс.
4. Высокая цена подписки – 15$ в месяц.
5. Нестабильная работа приложения.

Notion приложение для заметок, включающая в себя таблицы, графики, календари, с возможностью изменения дизайна. Notion существует под все операционные системы. Окно работы приложения предоставлено на рисунке 3.

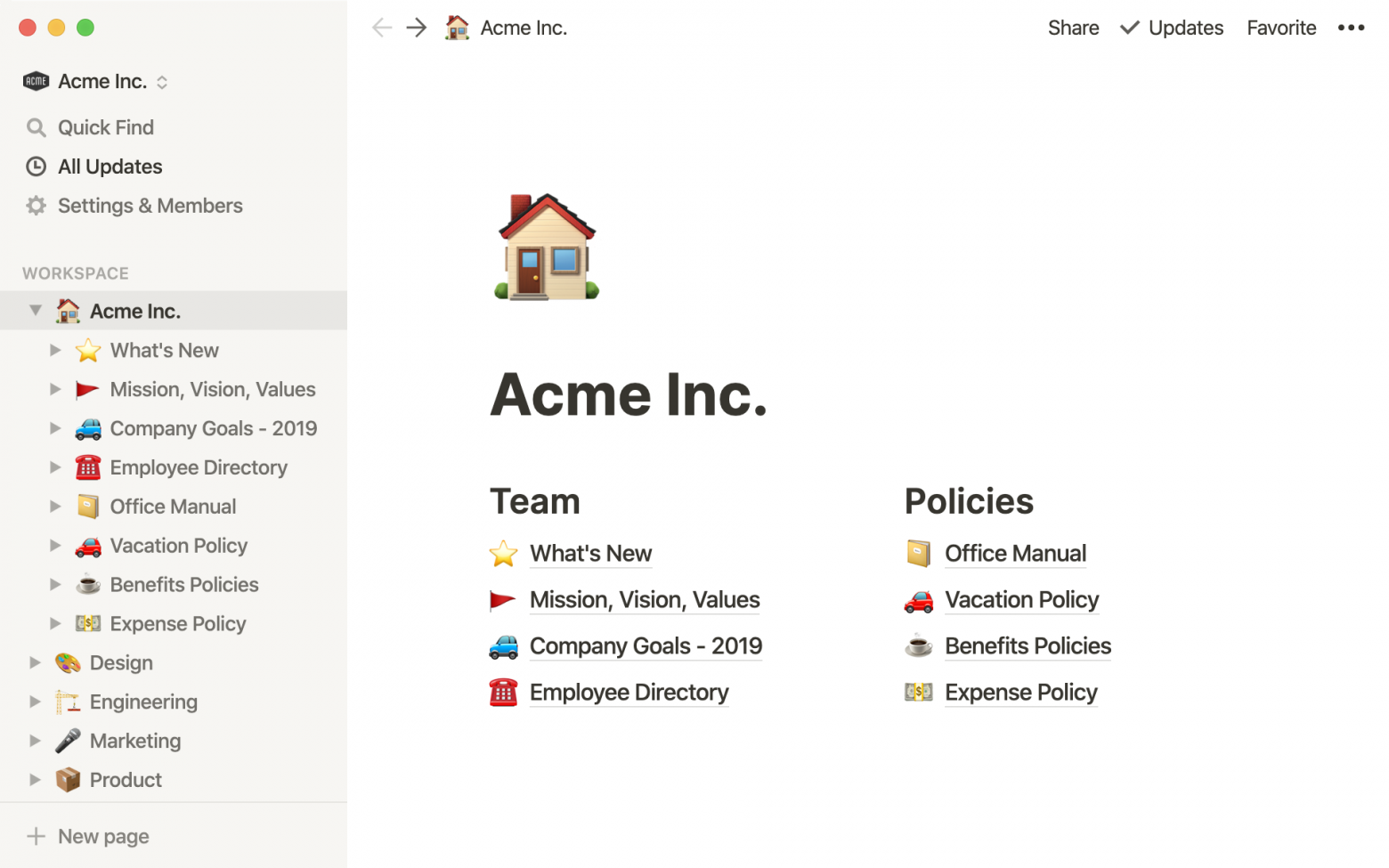


Рисунок 3 – окно приложения Notion

Минусы решения:

Приложение ещё находится в разработке, многие функции не работают

Сложная боковая панель

Рассмотрев данные приложения, можно сформировать требования, описанные в техническом задании.

* 1. Техническое задание

1. Общие сведения.

Требуется разработать мобильное приложение «Органайзер» для устройств под управлением Android. Иконка приложения должна быть выполнена в векторном стиле и не иметь лишних отвлекающих деталей. Приложение должно иметь возможность добавления и сохранения новой задачи, а также её корректирования. Весь интерфейс приложения должен быть на русском языке, допускается использование различных шрифтов и различных стилей написания. Приложение нужно предоставить до 26 апреля 2022 года с полным набором ниже перечисленных функций, и предоставлением отчётов о тестировании.

При открытии приложения, должна отображаться главная форма, в которой храниться весь список задач и дата.

При создании задачи, должна появляться функциональная возможность добавления названия задачи, её описания, добавлении уведомления, с возможностью указания точной даты и времени, и функционал по выбору иконки уведомления.

После создания приложения должна быть возможность её сохранения. После сохранения задача появляется в списке задач.

При нажатии на каждую задачу, должна появляться форма с её редактированием. Форма должна предоставлять возможность изменения любого поля.

В общем списке задач необходим функционал, позволяющий удалить любую задачу, без нарушения структуры приложения.

Приложение должна хранить задачу любое время и иметь защиту данных от повреждений.

1. Требования к системе.

Приложение должно быть доступно для устройств под управлением android, начиная с android версии 5.0 и иметь оптимизацию под устройства с большим размером экрана. Приложение необходимо оптимизировать, чтобы оно могло запускаться на устройстве любой мощности. Вес приложения допускается не более 7 Мб.

1. Состав и содержание работ по созданию системы.

Приложение должно быть разбито на несколько составляющих модулей, для более удобной дальнейшей работы с ним. Необходимо реализовать базу данных для хранения задач, модули работы с ней, а также модуль загрузи данных из базы в главное меню. Приложение должно иметь понятные комментарии на русском языке для каждого класса и функции.

1. Порядок контроля и приемки системы.

После реализации приложения и составления всей документации, проект требуется выложить на репозиторий GitHub для её проверки руководителем. Требуется отправить все исходные файлы, документацию и готовый apk файл приложения.

1. Требования к составу и содержанию работ по подготовке объекта к вводу системы в действие.

Требуется составить отчёты тестирования приложения, а также сопроводительную записку программиста и записку пользователя, с указанием принципов работы с приложением.

* 1. Описание структуры приложения

Приложение состоит из нескольких графических и программных модулей.

К графическим модулям относятся 3 activity:

Главная activity, которая будет отображаться при загрузке приложения. На данной activity будет отображаться все существующие записи, а также настоящая дата устройства.

Activity, позволяющая редактировать существующие или создавать новые записи. На данной activity будут отображаться все параметры записи, и поля для их изменения. Она должна содержать следующие параметры: название записи, описание, выбор иконки, добавление уведомления и выбор времени для уведомления.

Activity, позволяющая строить интерфейс. Это будет конкретный блок, который будет отображать определённую запись. Из этих блоков будет составляться список всех записей. Этот модуль должен отображать определённые параметры: название записи, иконку записи и в случае если запись имеет уведомление, время и дату, в которое придёт уведомление.

Программные модули приложения делятся на: модули для работы с базой данных и модуль работы с уведомлениями и модули для общей логики программы.

Модули работы с базой данных должны создавать базу, в случае если её нет на устройстве, обновлять её и импортировать данные о записях в проект.

В базе данных существуют следующие поля объекта: id записи, название, описание, иконка, поле, отвечающее за наличие уведомления и дату этого уведомления.

Модуль для работы с уведомлениями составляет структуру уведомления и выстраивает график их воспроизведения.

UML-диаграмма проекта изображена на рисунке 4.

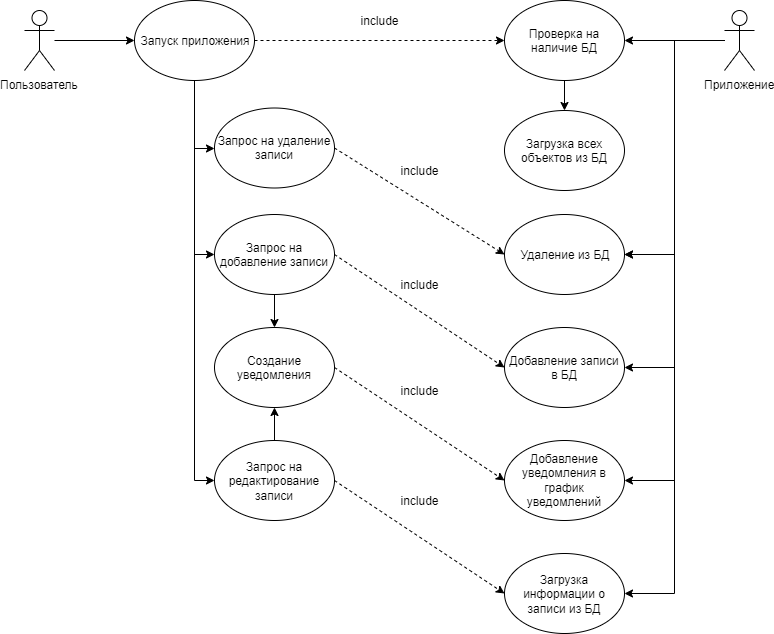


Рисунок 4 – UML-диаграмма проекта

1. Описание разработки приложения

2.1 Обоснование средств разработки

Для выполнения проекта необходимо определить, с помощью каких средств будет разрабатываться приложение. Самими популярными средствами разработки для мобильных являются: Android Studio [4], Xamarin [5] и React Native [6]. Требуется определить преимущества каждого из средств разработки, и исходя из технических требований выбрать самый подходящий вариант.

Android Studio – это IDE (интегрированная среда разработки) для работы с платформой Android. Она позволяет разрабатывать приложения для смартфонов, планшетов, часов, телевизоров и других устройств на этой ОС. Преимущества данного средства разработки:

* редактор кода, с которым удобно работать;
* Android Studio – это официальная среда разработки, которую поддерживает Google;
* позволяет разрабатывать приложения не только для смартфонов/планшетов, а и для портативных ПК, приставок для телевизоров Android TV, устройств Android Wear;
* тестирование корректности работы новых игр, утилит, их производительности на той или иной системе, происходит непосредственно в эмуляторе;
* рефакторинг уже готового кода;
* достаточно большая библиотека с готовыми шаблонами и компонентами для разработки ПО;
* разработка приложений для последних версий операционной системы Android;
* предварительная проверка уже созданного приложения на предмет ошибок в нем;
* большой набор средств инструментов для тестирования каждого элемента приложения, игры.

Недостатки Android Studio

* для тестирования приложения средствами встроенного Android-эмулятора требуется высокая аппаратная производительность ПК;
* невозможность написания серверных проектов на языке Java для ПК и Android устройств.

Xamarin – это платформа с открытым исходным кодом, предназначенная для построения современных производительных приложений для iOS, Android и Windows с .NET. Платформа Xamarin представляет собой уровень абстракции, который обеспечивает управление взаимодействием между общим кодом и кодом базовой платформы. Xamarin выполняется в управляемой среде, которая реализует такие возможности, как выделение памяти и сборка мусора.

Преимущества данного средства разработки:

* единый стек технологий для разработки на всех платформах;
* производительность близка к нативной;
* нативный UI;
* совместимость c оборудованием;
* Open Source-технологии с корпоративной поддержкой;
* благодаря кроссплатформенности, Xamarin упрощает поддержку и обновление программного обеспечения;
* полный пакет инструментов разработки.

Минусы Xamarin:

* задержки с обновлениями платформ;
* ограниченный доступ к open-source библиотекам;
* Xamarin не подходит для приложений с высокопроизводительной графикой;
* больший размер приложений;
* сложности с интеграцией.

React Native — это кроссплатформенный фреймворк с открытым исходным кодом для разработки нативных мобильных и настольных приложений на JavaScript и TypeScript, созданный Facebook, Inc. React Native поддерживает такие платформы как Android, Android TV, iOS, macOS, Apple tvOS, Web, Windows и UWP, позволяя разработчикам использовать возможности библиотеки React вне браузера для создания нативных приложений, имеющих полный доступ к системным API платформ.

Преимущества данного средства разработки:

* открытый исходный код;
* скорость и эффективность написания кода;
* функция горячей перезагрузки;
* высокая производительность;
* нативный UI.

Недостатки React Native:

* проект находиться в состоянии бета-версии;
* проблемы отладки и совместимости;
* зависимость от Facebook;
* низкий уровень управления памятью;
* проблемы с лицензией и патентом.

Исходя из выдвинутых требований к программному обеспечению, указанных в техническом задании, можно составить следующие требования к средству разработки:

* малый размер полученного программного проекта;
* упор на разработку программных продуктов для устройств под управлением Android;
* средство разработки должно позволять разрабатывать продукты под все версии Android, начиная с версии 5.0;
* возможность легкого и быстрого тестирования программного продукта;
* большой набор библиотек;
* высокая стабильность средства разработки;
* высокая поддержка со стороны разработчика средства.

Анализ имеющихся средств разработки позволяет сделать вывод, что лучшим решением для данного проекта является Android Studio, так как именно данное решение больше всего соответствует требования.

Android Studio предоставляет возможность двух разных способов разработки – средством языка программирования Java и Kotlin. Необходимо определить какой способ является наиболее предпочтительным для данного проекта. Для этого нужно определить преимущества и недостатки этих способов.

Преимущества языка программирования Java:

* С помощью виртуальной машины Java программы на этом языке могут запускаться практически в любой системе. В результате Java получает лидерство среди Android-приложений;
* Огромное сообщество;
* Кроссплатформенность;
* Поскольку Android разработан на Java, этот язык имеет уже готовые библиотеки и SDK для облегчения процесса разработки.

Недостатки Java:

* Java требует больше памяти и, по сравнению с другими языками, работает намного медленнее;
* На Java тяжелее писать код.

Преимущества языка программирования Kotlin:

* Повышенная производительность, достигающийся сокращением строк кода и оптимизацией;
* Полная совместимость с Java;
* Официальная поддержка Android Studio;
* Простота синтаксиса.

Недостатки Kotlin:

* Низкая скорость компиляции. В среднем компиляция на котлин дольше, чем на джава на 30-40 %. Это связано с тем, что компиляция в виртуальной машине JVM предназначена для джава, поэтому с родным языком она работает быстрее;
* Мало дополнений. В силу молодости языка у него намного меньше библиотек и фреймворков, чем у Java;
* Малочисленное сообщество.

Исходя из сравнения, можно сделать вывод, что язык программирования Java более прогнозируемый и стабильный по сравнению с Kotlin, а также имеет большую поддержку со стороны разработчиков. Поэтому в проекте будет использоваться средство разработки Android Studio с языком программирования Java.

Следующим этапом будет выбор базы данных. Самые популярные реляционные базы данных – MySQL, PostgreSQL, и SQLite. Для определения наиболее подходящего варианта, необходимо определить основные преимущества и недостатки каждого из них.

MySQL – это самая популярная реляционная база данных с открытым исходным кодом.

Платформа: Linux, Windows и Mac.

Языки: C, Java, SQL, C ++, Perl, Python, Tcl и PHP.

Преимущества:

* легко использовать;
* предоставляет большой функционал;
* хорошие функции безопасности;
* легко масштабируется и подходит для больших баз данных;
* обеспечивает хорошую скорость и производительность;
* обеспечивает хорошее управление пользователями и множественный контроль доступа.

Недостатки:

К недостаткам этой СУБД относятся:

* недостаточная надежность;
* низкая скорость разработки;
* отсутствие платформы Android.

PostgreSQL – это система управления базами данных с открытым исходным кодом корпоративного класса, ориентирующаяся в первую очередь на полное соответствие стандартам и расширяемость. Он поддерживает как SQL для реляционных, так и JSON для нереляционных запросов.

PostgreSQL отличается от других РСУБД тем, что обладает объектно-ориентированным функционалом, в том числе полной поддержкой концепта ACID (Atomicity, Consistency, Isolation, Durability).

Платформа: Mac, Windows и Linux.

Языки: PL / Tcl, PL / pgSQL, PL / Python и PL / Perl.

Преимущества:

* Полная SQL-совместимость.
* Расширяемость. PostgreSQL можно программно расширить за счёт хранимых процедур;
* Объектно-ориентированность. PostgreSQL — не только реляционная, но и объектно-ориентированная СУБД.

Недостатки:

* Производительность. В простых операциях чтения PostgreSQL может уступать своим соперникам;
* Популярность. Из-за своей сложности инструмент не очень популярен;

SQLite — это библиотека, встраиваемая в приложение, которое её использует. Являясь файловой БД, она предоставляет большой набор инструментов для более простой обработки любых видов данных.

Когда приложение использует SQLite, их связь производится с помощью функциональных и прямых вызовов файлов, содержащих данные, что повышает скорость и производительность операций.

Преимущества:

* Файловая: вся база данных хранится в одном файле, что облегчает перемещение;
* Стандартизированная: SQLite использует SQL; некоторые функции опущены, однако, есть и некоторые новые;
* Отлично подходит для разработки и даже тестирования.

Недостатки:

* Отсутствие пользовательского управления: продвинутые БД предоставляют пользователям возможность управлять связями в таблицах в соответствии с привилегиями, но у SQLite такой функции нет;
* Невозможность дополнительной настройки.

В связи с тем, что приложение разрабатывается для мобильного устройства, нет необходимости в большом функционале. В базе данных приложения не будет множества связей и таблиц. Поэтому требуется простая база данных с удобной и быстрой разработкой. Больше всего по этим требованиям подходит библиотека SQLite, с которой можно быстро и качественно разработать необходимую структуру базы.

2.2 Разработка пользовательского интерфейса

Исходя из описания структуры проекта, в котором определены три графических модуля: главная activity, activity редактирования и activity блока записи, необходимо разработать пользовательский интерфейс.

Первой следует разработать блок записи под названием item\_list\_layout. Макет item\_list\_layout изображен на рисунке 5. В нем будут размещены следующие объекты:

* Картинка, располагающаяся в левой части блока;
* Название, привязанное к картинке, находящееся левее её;
* Время уведомления, находящееся с левой части блока.

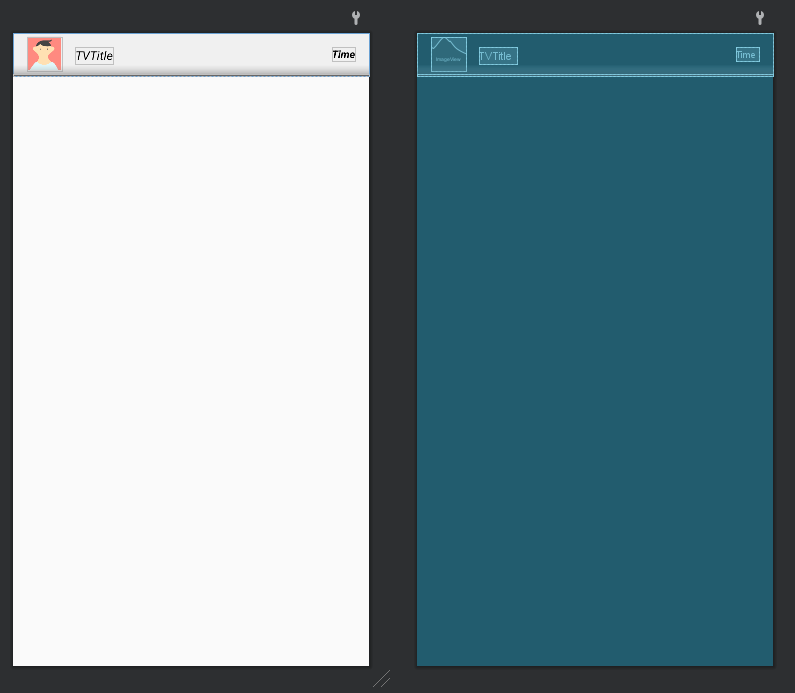


Рисунок 5 – Макет блока записи item\_list\_layout

Следом разрабатывается activity редактирования – activity\_edit. Макет activity\_edit изображен на рисунке 6. В ней размещается следующие объекты:

* Шапка, в которой отображается одно тестовое поле с надписью: «Работа с заметками»;
* Главный блок редактирования и создания, располагающийся в центре экрана. Это вертикальный layout, в котором расположены: поле ввода названия заметки, поле добавления уведомления с флажком, выбор иконки с помощью выпадающего меню, текстовое поле с кнопкой для открытия меню выбора даты и многострочный блок ввода для описания заметки;
* Кнопка сохранения, располагающееся с нижней части экрана по центру.

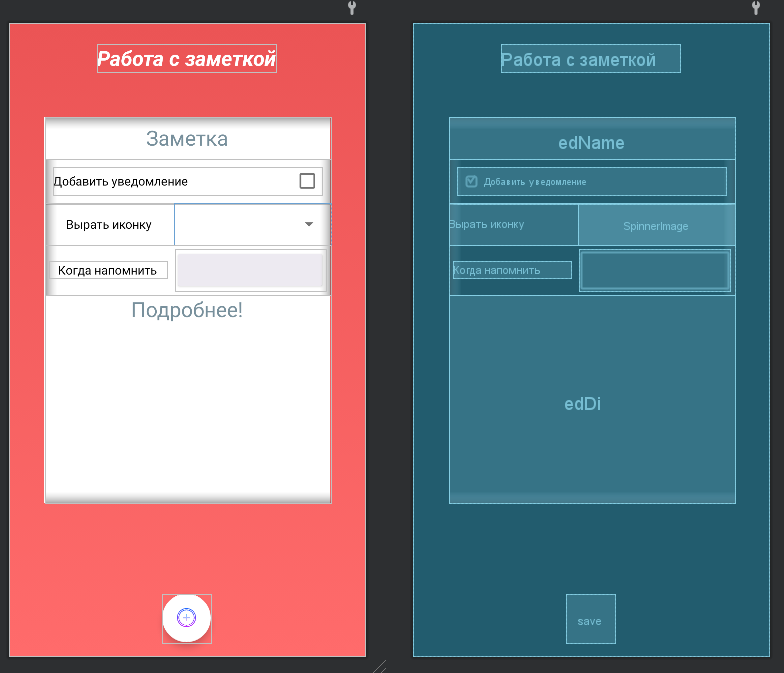


Рисунок 6 – Макет activity редактирования activity\_edit

Последней следует разработать главную activity под названием activity\_main. Макет activity\_main изображен на рисунке 7. В ней будут размещены следующие объекты:

* шапка, в которой отображаются 4 текстовых поля – год, месяц, число и день недели устройства;
* основная часть, в которой отображается список объектов item\_list\_layout;
* кнопка, для открытия activity\_edit, отображающаяся в правом нижнем углу экрана.

Макет main\_activity

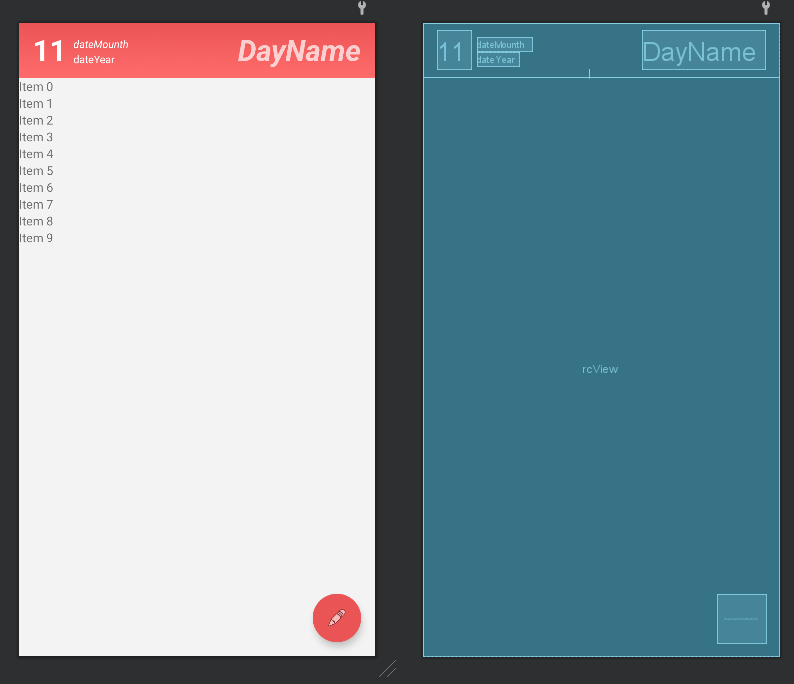


Рисунок 7 – Макет activity редактирования activity\_main

2.3 Разработка логики работы приложения. Схема взаимодействия компонентов проекта.

Для написания программы необходимо составить связь модулей приложения. Все модули взаимосвязаны друг с другом, поэтому нужно правильно выстроить структуру, для корректной работы приложения. Схема взаимодействия компонентов проекта изображена на рисунке 8.

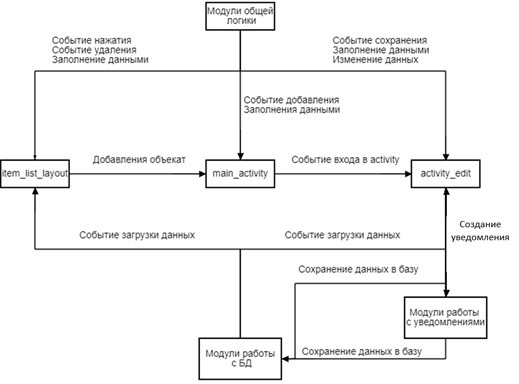


Рисунок 8 – Схема взаимодействия компонентов проекта

2.4 Описание переменных, компонентов, классов и подпрограмм

Классы adapter – классы, созданные для получения записей из базы данных.

Класс Link позволяет присвоить переменным поля заметки из базы данных. Класс состоит из полей (таблица 1), методов, возвращающих значения – геттеров (таблица 2) и методов, задающих значения – сеттеров (таблица 3).

Таблица 1 – Поля класса Link

|  |  |
| --- | --- |
| Название | Назначение |
| Title | Хранит название заметки |
| Desc | Хранит описание заметки |
| Date | Хранит дату уведомления заметки |
| Image | Хранит номер картинки заметки |
| Notify | Хранит наличие уведомления у заметки |
| Id | Хранит уникальный номер заметки |

Таблица 2 – методы геттеры класса Link

|  |  |
| --- | --- |
| Название | Назначение |
| getId | Возвращает уникальный номер заметки |
| getTitle | Возвращает название заметки |
| getDesc | Возвращает описание заметки |
| getDate | Возвращает дату уведомления заметки |
| getImage | Возвращает номер картинки заметки |
| getNotify | Возвращает наличие уведомления заметки |

Таблица 3 – Методы сеттеры класса Link

|  |  |
| --- | --- |
| Название | Назначение |
| setId | Задаёт уникальный номер заметки |
| setTitle | Задаёт название заметки |
| setDesc | Задаёт описание заметки |
| setDate | Задаёт дату уведомления заметки |
| setImage | Задаёт номер картинки заметки |
| setNotify | Задаёт наличие уведомления заметки |

Класс MainAdapter позволяет получать записи из базы данных и заполнять ими массив. Класс состоит из полей (таблица 4) и методов (таблица 5).

Таблица 4 – Поля класса MainAdapter

|  |  |  |
| --- | --- | --- |
| Название | Назначение | Значение |
| Context | Хранит контекст приложения, позволяющий определить, в какой activity выполняется код | None |
| Array | Список заметок, пришедших из базы данных | new ArrayList<>() |
| Tv | Объект, хранящий текстовое поле названия заметки. Нужен для записи в него значения | itemView.findViewById(R.id.TVTitle) |

Продолжение таблицы 4

|  |  |  |
| --- | --- | --- |
| Td | Объект, хранящий текстовое поле описания заметки. Нужен для записи в него значения | itemView.findViewById(R.id.Time) |
| Ti | Объект, хранящий поле картинки заметки. Нужен для установки картинки. | itemView.findViewById(R.id.imageView) |

Таблица 5 – Методы класса MainAdapter

|  |  |  |
| --- | --- | --- |
| Название | Назначение | Принимающиеся поля |
| MainAdapter | Конструктор класса, нужен для присвоения данных полям. | Контекст приложения |
| onCreateViewHolder | Создаёт обёртку списка задач | Лист со всеми записями, значение типа состояния |
| onBindHolder | Заполняет список задач объектами из массива | Объект класса MyViewHolder, индекс массива записи |
| getItemCount | Получает количество всех объектов в массиве | None |
| upAdapter | Очищает и заполняет массив записей новыми элементами | Лист со всеми записями |
| Remote | Очищает массив записей. | индекс массива записи, объект класса DBManager |

Подкласс MyViewHolder – класс для получения данных из массива и заполнения ими полей формы. Класс состоит из методов (таблица 6).

Таблица 6 – Методы класса MyViewHolder

|  |  |  |
| --- | --- | --- |
| Название | Назначение | Принимающиеся поля |
| MyViewHolder | Конструктор, записывающий в переменные объекты формы | Activity с её объектами, контекст приложения, массив элементов |

Продолжение таблицы 6

|  |  |  |
| --- | --- | --- |
| setData | Заполняет объекты данными | Переменные названия, даты и номер картинки записи |
| onClick | Передаёт в форму редактирования выбранный объект. | Activity с её объектами |

Классы db – набор классов для создания структуры базы данных и добавления в таблицу значений. Класс состоит из полей (таблица 7).

Класс Constant – класс для хранения констант базы данных

Таблица 7 – Поля класса Constant

|  |  |  |
| --- | --- | --- |
| Название | Назначение | Значение |
| TABLE\_NAME | Поле для хранения названия таблицы базы данных | task\_tabele |
| LIST\_ITEM\_INTENT | Поле для хранения содержимого записи | list\_item\_intent |
| EDIT\_STATE | Поле для хранения операции редактирования | edit\_state |
| ID | Поле для хранения названия уникального номера | id |
| TITLE | Поле для хранения названия заголовка заметки | title |
| DISC | Поле для хранения названия описания заметки | disc |
| DATE | Поле для хранения названия даты заметки | date |
| IMAGE | Поле для хранения названия картинки заметки | image |
| NOTIFY | Поле для хранения названия наличия уведомления заметки | notify |
| DB\_NAME | Поле для хранения названия базы данных | my\_db.db |
| DB\_VERSION | Поле для хранения версии базы данных | 1 |

Продолжение таблицы 7

|  |  |  |
| --- | --- | --- |
| TABLE\_STRUCTURE | Поле для хранения запроса на создания таблицы | CREATE TABLE IF NOT EXISTS " + TABLE\_NAME + " (" + ID + " INTEGER PRIMARY KEY," + TITLE + " TEXT, "+ DISC + " TEXT, "+ DATE + " TEXT, " + IMAGE + " TEXT," + NOTIFI+ " INTEGER |
| DROP\_TABEL | Поле для хранения запроса на удаление таблицы | DROP TABLE IF EXISTS " + TABLE\_NAME |

Класс DBHelper – возвращает имеющуюся базу данных или создаёт новую при её отсутствии. Класс состоит из методов (таблица 8).

Таблица 8 – Методы класса DBHelper

|  |  |  |
| --- | --- | --- |
| Название | Назначение | Принимающиеся поля |
| DBHelper | Конструктор, записывающий в имеющуюся базу данных | Контекст приложения |
| onCreate | Метод создающий базу данных | База данных |
| onUpgrade | Метод обновляющий базу данных | База данных, номер предыдущий версии, номер новой версии |

Класс DBManager – класс, позволяющий обновить базу данных, добавить в неё значение и получить из неё поля. Класс состоит из методов (таблица 9).

Таблица 9 – Методы класса DBManager

|  |  |  |
| --- | --- | --- |
| Название | Назначение | Принимающиеся поля |
| DBManager | Конструктор, создающий объект DBHelper исходя из контекста | Контекст приложения |
| OpenDB | Открывает соединение с базой данных | None |
| delete | Удаляет запись из базы данных по её уникальному номеру | Уникальный номер записи |
| update | Метод, обновляющий заметку в базе данных | Название, описание, дату, номер картинки, наличие уведомления и уникальный номер записи |
| insertToDB | Метод, добавляющий в базу данных запись | Название, описание, дату, номер картинки и наличие уведомления записи |
| getFromDb | Метод, заполняющий массив записей | None |
| closeDB | Метод, закрывающий соединение с базой данных | None |

Reminder – класс, создающий объект уведомления и добавляющий его в массив. Класс состоит из полей (таблица 10) и методов (таблица 11).

Таблица 10 – Поля класса Reminder

|  |  |  |
| --- | --- | --- |
| Название | Назначение | Стандартные значения |
| Reminder | Массив хранящий объекты уведомлений | none |
| arrayid | Хранит уникальный номер выбранного объекта уведомления | 0 |
| title | Хранит название уведомления | У вас важное уведомление! |
| desc | Хранит описание уведомления | Зайдите в приложение |
| id | Хранит уникальный номер уведомления | 1 |
| image | Хранит номер уведомления | 1 |

Продолжение таблицы 10

|  |  |  |
| --- | --- | --- |
| date | Хранит дату, когда придёт уведомление | 0 |

Таблица 11 – Методы класса Reminder

|  |  |  |
| --- | --- | --- |
| Название | Назначение | Принимающиеся поля |
| create | Создаёт и добавляет объект класса Reminder | Название, описание, номер картинки, уникальный номер и дату уведомления |
| onReceive | Создаёт уведомление и добавляет его в канал уведомлений | Контекст приложения и содержимое контекста |
| first | Возвращает первый объект массива уведомлений | None |
| sort | Сортирует массив уведомлений по дате | None |

Loading – класс activity Loading. Открывает канал уведомлений при открытии приложения. Класс состоит из методов (таблица 12).

Таблица 12 – Методы класса Loading

|  |  |
| --- | --- |
| Название | Назначение |
| onCreate | Инициализирует объекты Activity |
| createNotificatinChannel | Создаёт объект класса NotificationManager и открывает канал уведомлений |

Класс Edit – класс activity Edit. Работает с записями базы данных, выводя их значения на экран, и принимающий значения из полей ввода. Класс состоит из методов (таблица 13).

Таблица 13 – Методы класса Edit

|  |  |
| --- | --- |
| Название | Назначение |
| onCreate | Инициализирует объекты Activity |
| onResume | Отрывает соединение с базой данных |

Продолжение таблицы 13

|  |  |
| --- | --- |
| onClickSave | Сохраняет или редактирует запись в базе данных, путём получения значений из полей ввода |
| setTime, setInitialDateTime, onTimeSet, onDateSet | Комплекс методов для работы с DatePickerDialog. Позволяет выбрать дату уведомления, путём выбора даты в всплывающий окнах |
| SendMess | Добавляет в поток уведомлений, объект класса Reminder |
| CheckBoxNotify | Метод, определяющий наличие уведомления у заметки |

Класс MainActivity – класс activity MainActivity. Работает с главным окном, позволяет работать со списком записей. Класс состоит из методов (таблица 14).

Таблица 14 – Методы класса MainActivity

|  |  |
| --- | --- |
| Название | Назначение |
| onCreate | Инициализирует объекты Activity |
| onResume | Отрывает соединение с базой данных и создаёт объект класса MainAdapter |
| onClickAdd | Открывает activity Edit для создания записи |
| onDestroy | Закрывает соединение с базой данных при закрытии приложения |
| getItemTouchHelper | Метод для удаления записи свайпом по экрану |

Схема взаимодействия классов изображена на рисунке 9.

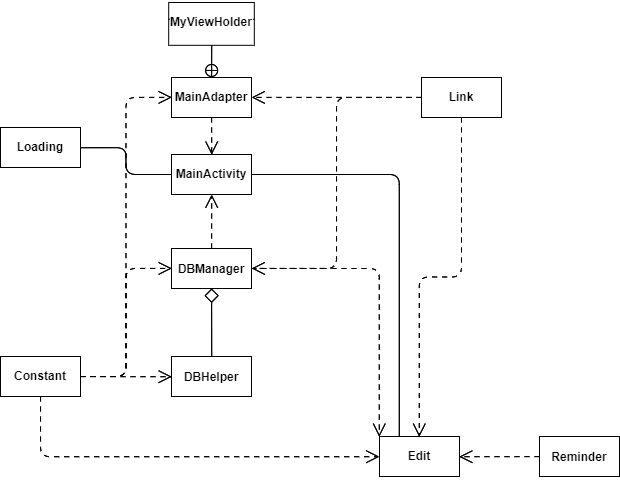


Рисунок 9 – Схема взаимодействия классов

1. Тестирование и установка приложения

После разработки приложения следует произвести установку и тестирование приложения.

Первым этапом нужно собрать приложение, как изображено на рисунке 10.

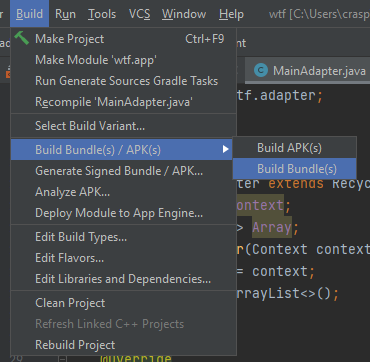


Рисунок 10 – Сборка проекта

Следующий этап – установить и запустить apk файл на телефоне. После этого произойдёт установка приложения на телефон.

Дальнейшими этапами будет функциональное и UI-тестирование.

Тест 1. Запуск приложения. Результат теста изображён на рисунке 11.

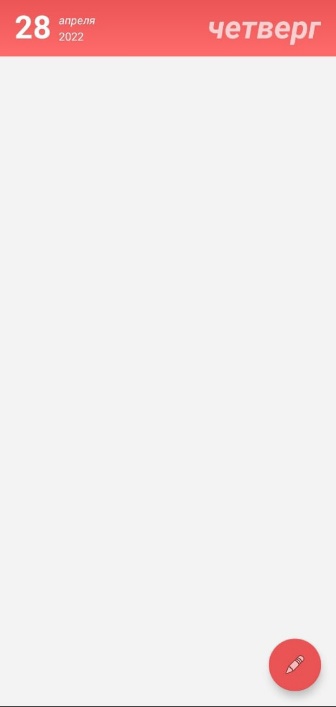


Рисунок 11 – Результат первого теста

Тест пройдет успешно. Приложение открылось и отобразило необходимые элементы.

Тест 2. Создание новой заметки. Результат теста изображён на рисунке 12.

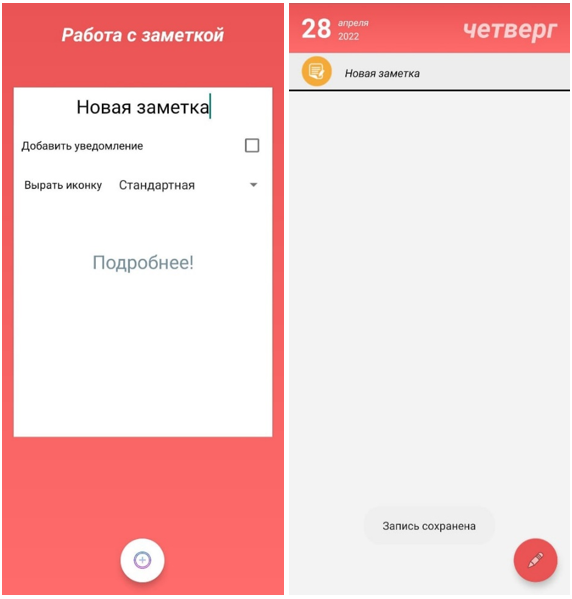


Рисунок 12 – Результат второго теста

Тест пройдет успешно. Запись добавилась в базу данных с введёнными параметрами.

Тест 3. Создание уведомления. Результат теста изображён на рисунке 13.

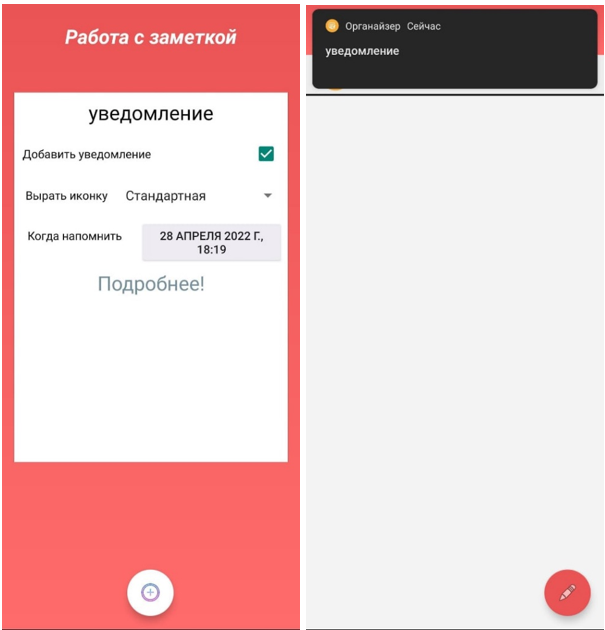


Рисунок 13 – Результат третьего теста

Тест пройдет успешно. Приложение записало в базу запись с уведомлением. Уведомление отобразилось по указанному времени, с введёнными параметрами.

Тест 4. Попытка перегрузки приложения большим количеством различных заметок. Тест проверяет не только перегрузку приложения, но и все функциональные настройки заметки. Результат теста изображён на рисунке 14.



Рисунок 14 – Результат четвёртого теста

Тест пройдет успешно. Все заметки записались и отобразились, при этом каждая запись имеет с уникальное значение.

Тест 5. Удаление заметки. Результат теста изображён на рисунке 15.

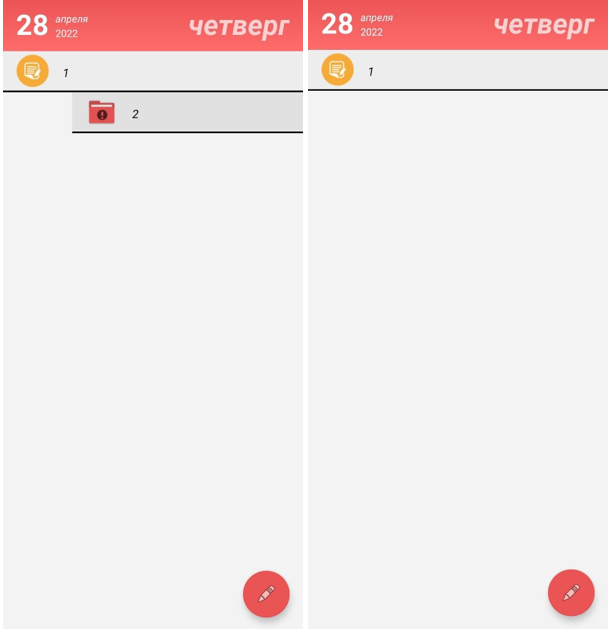


Рисунок 15 – Результат пятого теста

Тест пройдет успешно. Запись пропала из интерфейса и исчезла из базы данных.

После проведение тестирования с включенной отладкой, ошибок в консоли не обнаружено (рисунок 16).

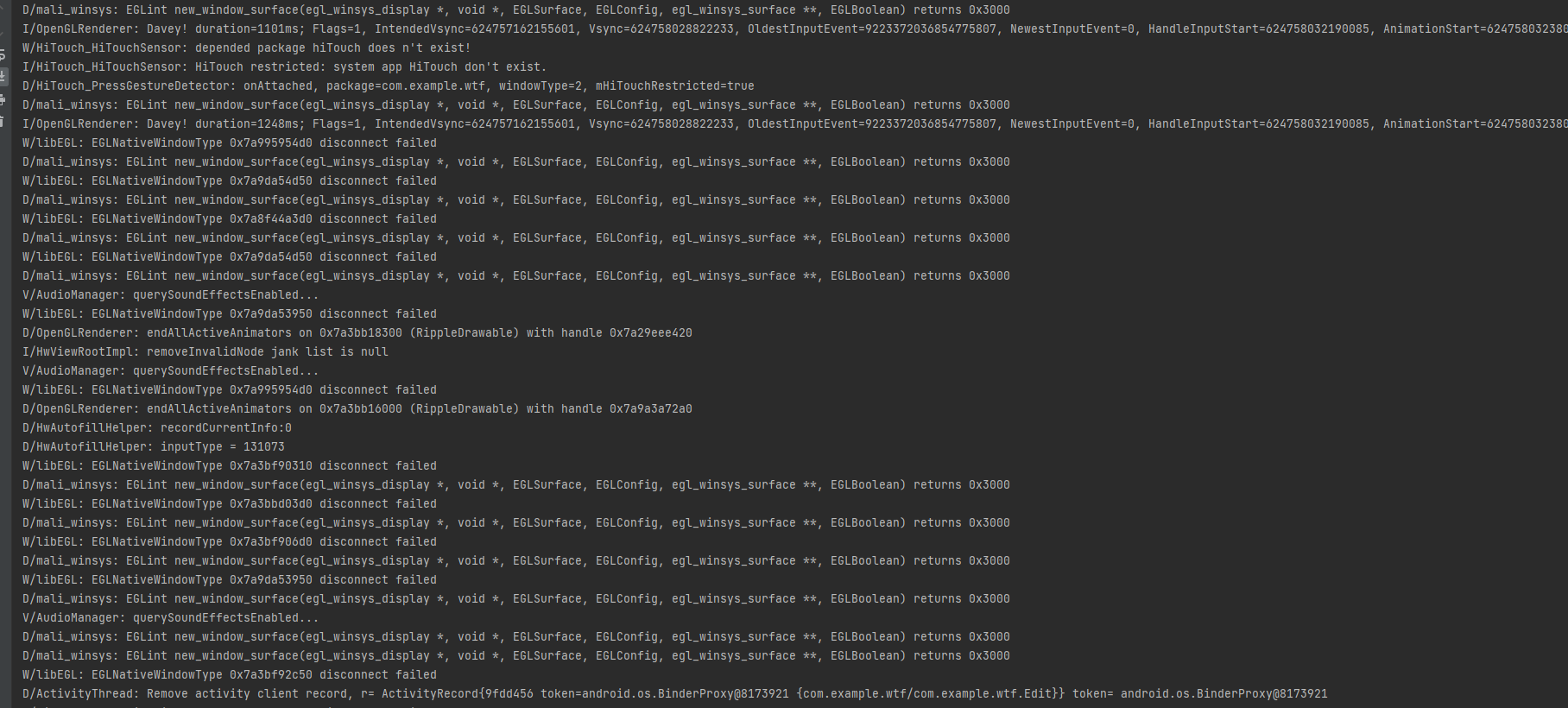


Рисунок 16 – Консоль отладки приложения

В результате тестирования ошибок не обнаружено.

Заключение

В ходе выполнения тестового проекта была проведена работа с анализом рынка. Проведено сравнение подобных приложений, рассмотрены их основные преимущества и недостатки, а также ключевые особенности. С помощью такого анализа стало возможным сформировать техническое задание, определились основные потребности рынка. После написания технического задания сформировались функциональные требования к приложению, из которых появилась необходимость в выборе средств разработки. Были разобраны наиболее популярные и подходящие решения и после продолжительного анализа, был произведён выбор.

Далее происходил процесс разработки программного обеспечения, разделённый на этап проектирования интерфейса и этап написания программного кода. После создания нескольких взаимосвязанных модулей и их компоновки друг с другом получилось полноценное мобильное приложение.

Последним этапом стали отладка и тестирование получившегося программного продукта, после прохождения которого, приложение стало доступно для его выпуска и использования пользователями.

В работе выполнены все поставленные задачи и цели.

Список использованной литературы

1. Приложение Taskade (https://www.taskade.com/).

2. Приложение Roam Research (https://roamresearch.com/).

3. Приложение Notion (https://www.notion.so/).

4. Средство разработки Android Studio (https://developer.android.com/studio).

5. Средство разработки Xamarin (https://dotnet.microsoft.com/en-us/apps/xamarin).

6. Средство разработки React Native (https://reactnative.dev/).

Приложение. Термины.

Модуль — функционально законченный фрагмент программы, оформленный в виде отдельного файла с исходным кодом. Модуль позволяет облегчить разработку и сопровождение программного продукта, а также увеличивает качество и скорость тестирования.

Activity – это компонент приложения, который является одним из его фундаментальных строительных блоков. Его основное предназначение заключается в том, что оно служит точкой входа для взаимодействия приложения с пользователем, а также отвечает за то, как пользователь перемещается внутри приложения или между приложениями.

Запись – это создаваемый или редактируемый компонент приложения. В данном приложении записью будет являться заметка, задача или идея, которые будет генерировать пользователь.

Метод – это блок кода, состоящий из ряда инструкций, который можно вызывать по его имени.

Поле – поименованная, либо адресуемая иным способом область памяти, адрес которой можно использовать для осуществления доступа к данным.

Приложение А. Листинг рабочего кода

Activity\_edit.xml

<?xml version="1.0" encoding="utf-8"?>

<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:app="http://schemas.android.com/apk/res-auto"

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:background="@drawable/gradient\_bg"

android:theme="@style/Theme.Design.Light.NoActionBar"

tools:context=".Edit">

<TextView

android:id="@+id/textView"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_marginTop="24dp"

android:text="Работа с заметкой"

android:textColor="#FFFFFF"

android:textSize="24sp"

android:textStyle="bold|italic"

app:layout\_constraintEnd\_toEndOf="parent"

app:layout\_constraintHorizontal\_bias="0.498"

app:layout\_constraintStart\_toStartOf="parent"

app:layout\_constraintTop\_toTopOf="parent" />

<LinearLayout

android:id="@+id/linearLayout2"

android:layout\_width="330dp"

android:layout\_height="446dp"

android:layout\_marginTop="52dp"

android:background="#FFFFFF"

android:orientation="vertical"

app:layout\_constraintEnd\_toEndOf="parent"

app:layout\_constraintStart\_toStartOf="parent"

app:layout\_constraintTop\_toBottomOf="@+id/textView">

<EditText

android:id="@+id/edName"

android:layout\_width="match\_parent"

android:layout\_height="48dp"

android:background="@color/white"

android:ems="10"

android:gravity="center"

android:hint="Заметка"

android:inputType="textPersonName"

android:textColor="@color/bg"

android:textColorHint="#78909C"

android:textSize="24sp" />

<LinearLayout

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:orientation="horizontal">

<CheckBox

android:id="@+id/NotifiCheckBox"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:layout\_margin="10dp"

android:layout\_weight="1"

android:button="@null"

android:drawableRight="?android:attr/listChoiceIndicatorMultiple"

android:onClick="CheckBoxNotify"

android:text="Добавить уведомление" />

</LinearLayout>

<LinearLayout

android:id="@+id/IconLayout"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:background="@color/white"

android:orientation="horizontal">

<TextView

android:id="@+id/textView2"

android:layout\_width="116dp"

android:layout\_height="match\_parent"

android:layout\_weight="1"

android:gravity="center"

android:text="Вырать иконку"

android:textColor="@color/bg" />

<Spinner

android:id="@+id/SpinnerImage"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_weight="4"

android:dropDownWidth="match\_parent"

android:dropDownSelector="@color/bg"

android:minHeight="48dp"

android:popupBackground="#F47D7D"

android:spinnerMode="dropdown"

android:textColor="@color/bg" />

</LinearLayout>

<LinearLayout

android:id="@+id/TimeLayout"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:orientation="horizontal">

<TextView

android:id="@+id/textView3"

android:layout\_width="136dp"

android:layout\_height="wrap\_content"

android:layout\_margin="5dp"

android:gravity="top|center"

android:text="Когда напомнить"

android:textColor="@color/bg" />

<Button

android:id="@+id/DataText"

android:layout\_width="162dp"

android:layout\_height="wrap\_content"

android:layout\_margin="5dp"

android:layout\_weight="1"

android:onClick="setTime"

app:backgroundTint="#EDEAF1" />

</LinearLayout>

<EditText

android:id="@+id/edDick"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:background="@color/white"

android:ems="10"

android:fadeScrollbars="true"

android:gravity="top|center"

android:hapticFeedbackEnabled="false"

android:hint="Подробнее!"

android:inputType="textMultiLine"

android:textColor="@color/bg"

android:textColorHint="#78909C"

android:textSize="24sp" />

</LinearLayout>

<com.google.android.material.floatingactionbutton.FloatingActionButton

android:id="@+id/save"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_marginBottom="16dp"

android:clickable="true"

android:onClick="onClickSave"

app:backgroundTint="#FFFFFF"

app:fabSize="normal"

app:layout\_constraintBottom\_toBottomOf="parent"

app:layout\_constraintEnd\_toEndOf="parent"

app:layout\_constraintHorizontal\_bias="0.498"

app:layout\_constraintStart\_toStartOf="parent"

app:srcCompat="@drawable/add"

tools:ignore="SpeakableTextPresentCheck" />

</androidx.constraintlayout.widget.ConstraintLayout>

Activity\_main.xml

<?xml version="1.0" encoding="utf-8"?>

<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:app="http://schemas.android.com/apk/res-auto"

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:background="#F4F3F3"

android:onClick="onClickSave"

android:theme="@style/Theme.Design.Light.NoActionBar"

tools:context=".MainActivity">

<androidx.recyclerview.widget.RecyclerView

android:id="@+id/rcView"

android:layout\_width="0dp"

android:layout\_height="0dp"

android:onClick="onClickAdd"

app:layout\_constraintBottom\_toBottomOf="parent"

app:layout\_constraintEnd\_toEndOf="parent"

app:layout\_constraintHorizontal\_bias="0.0"

app:layout\_constraintStart\_toStartOf="parent"

app:layout\_constraintTop\_toBottomOf="@+id/constraintLayout" />

<com.google.android.material.floatingactionbutton.FloatingActionButton

android:id="@+id/floatingActionButton2"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_marginEnd="16dp"

android:layout\_marginRight="16dp"

android:layout\_marginBottom="16dp"

android:background="#007C4DFF"

android:clickable="true"

android:onClick="onClickAdd"

app:backgroundTint="@color/dark\_main"

app:layout\_constraintBottom\_toBottomOf="parent"

app:layout\_constraintEnd\_toEndOf="parent"

app:srcCompat="@android:drawable/ic\_menu\_edit" />

<androidx.constraintlayout.widget.ConstraintLayout

android:id="@+id/constraintLayout"

android:layout\_width="0dp"

android:layout\_height="wrap\_content"

android:background="@drawable/gradient\_bg"

android:theme="@style/Theme.Design"

app:layout\_constraintEnd\_toEndOf="parent"

app:layout\_constraintStart\_toStartOf="parent"

app:layout\_constraintTop\_toTopOf="parent">

<TextView

android:id="@+id/empty"

android:layout\_width="wrap\_content"

android:layout\_height="10dp"

android:layout\_marginStart="192dp"

android:layout\_marginLeft="192dp"

android:background="#00536DFE"

android:gravity="left|center"

android:textColor="@color/white"

android:textSize="20sp"

app:layout\_constraintStart\_toStartOf="parent"

app:layout\_constraintTop\_toBottomOf="@+id/dateDay" />

<TextView

android:id="@+id/dateYear"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_marginStart="8dp"

android:layout\_marginLeft="8dp"

android:layout\_marginTop="1dp"

android:background="#00536DFE"

android:gravity="left|center"

android:text="dateYear"

android:textColor="@color/white"

android:textSize="12sp"

app:layout\_constraintStart\_toEndOf="@+id/dateDay"

app:layout\_constraintTop\_toBottomOf="@+id/dateMounth" />

<TextView

android:id="@+id/dateMounth"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_marginStart="8dp"

android:layout\_marginLeft="8dp"

android:layout\_marginTop="16dp"

android:background="#00536DFE"

android:gravity="left|center"

android:text="dateMounth"

android:textColor="@color/white"

android:textSize="12sp"

android:textStyle="italic"

app:layout\_constraintStart\_toEndOf="@+id/dateDay"

app:layout\_constraintTop\_toTopOf="parent" />

<TextView

android:id="@+id/dateDay"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_marginStart="16dp"

android:layout\_marginLeft="16dp"

android:layout\_marginTop="8dp"

android:background="#00536DFE"

android:gravity="left|center"

android:text="11"

android:textColor="@color/white"

android:textSize="34sp"

android:textStyle="bold"

app:layout\_constraintStart\_toStartOf="parent"

app:layout\_constraintTop\_toTopOf="parent" />

<TextView

android:id="@+id/DayName"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_marginTop="8dp"

android:layout\_marginEnd="16dp"

android:layout\_marginRight="16dp"

android:text="DayName"

android:textSize="34sp"

android:textStyle="bold|italic"

app:layout\_constraintEnd\_toEndOf="parent"

app:layout\_constraintTop\_toTopOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>

</androidx.constraintlayout.widget.ConstraintLayout>

Item\_list\_layout.xml

<?xml version="1.0" encoding="utf-8"?>

<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:app="http://schemas.android.com/apk/res-auto"

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:background="#99EAE9E9"

android:theme="@style/Theme.Design.Light.NoActionBar">

<TextView

android:id="@+id/TVTitle"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_marginStart="16dp"

android:layout\_marginLeft="16dp"

android:layout\_marginTop="16dp"

android:background="#00536DFE"

android:gravity="left|center"

android:text="TVTitle"

android:textColor="@color/bg"

android:textStyle="italic"

app:layout\_constraintStart\_toEndOf="@+id/imageView"

app:layout\_constraintTop\_toTopOf="parent" />

<TextView

android:id="@+id/Time"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_marginTop="16dp"

android:layout\_marginEnd="16dp"

android:layout\_marginRight="16dp"

android:background="#00536DFE"

android:gravity="left|center"

android:text="Time"

android:textColor="@color/bg"

android:textSize="12sp"

android:textStyle="bold|italic"

app:layout\_constraintEnd\_toEndOf="parent"

app:layout\_constraintTop\_toTopOf="parent" />

<ImageView

android:id="@+id/imageView"

android:layout\_width="40dp"

android:layout\_height="39dp"

android:layout\_marginStart="16dp"

android:layout\_marginLeft="16dp"

android:layout\_marginTop="4dp"

app:layout\_constraintStart\_toStartOf="parent"

app:layout\_constraintTop\_toTopOf="parent"

tools:srcCompat="@tools:sample/avatars" />

<LinearLayout

android:id="@+id/line"

android:layout\_width="match\_parent"

android:layout\_height="2dp"

android:layout\_marginTop="4dp"

android:background="@color/bg"

android:orientation="vertical"

app:layout\_constraintEnd\_toEndOf="parent"

app:layout\_constraintStart\_toStartOf="parent"

app:layout\_constraintTop\_toBottomOf="@+id/imageView">

</LinearLayout>

</androidx.constraintlayout.widget.ConstraintLayout>

Link.java

package com.example.wtf.adapter;

import java.io.Serializable;

public class link implements Serializable {

private String title;

private String desc;

private String date;

private String image;

private int notifi;

private int id = 0;

public int getId() {return id;}

public void setId(int id) {this.id = id; }

public String getTitle() {return title; }

public void setTitle(String title) {this.title = title;}

public String getDesc() {return desc; }

public void setDesc(String desc) { this.desc = desc;}

public String getDate() {return date; }

public void setDate(String date) {this.date = date;}

public String getImage() {return image; }

public void setImage(String image) {this.image = image;}

public boolean getNotifi() {

if (notifi == 1){ return true; }

else {return false; }}

public void setNotifi(int notifi) {this.notifi = notifi; }}

MainAdapter.java

package com.example.wtf.adapter;

import android.content.Context;

import android.content.Intent;

import android.view.LayoutInflater;

import android.view.View;

import android.view.ViewGroup;

import android.widget.ImageView;

import android.widget.TextView;

import androidx.annotation.NonNull;

import androidx.recyclerview.widget.RecyclerView;

import com.example.wtf.Edit;

import com.example.wtf.R;

import com.example.wtf.db.Constant;

import com.example.wtf.db.DBManager;

import java.util.ArrayList;

import java.util.List;

public class MainAdapter extends RecyclerView.Adapter<MainAdapter.MyViewHolder> {

private Context context;

private List<link> Array;

public MainAdapter(Context context){

this.context = context;

Array = new ArrayList<>();}

@NonNull

@Override

public MyViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {

View view = LayoutInflater.from(context).inflate(R.layout.item\_list\_layout, parent,false);

return new MyViewHolder(view, context, Array); }

@Override

public void onBindViewHolder(@NonNull MyViewHolder holder, int position) {

holder.setData(Array.get(position).getTitle(), Array.get(position).getDate(), Array.get(position).getImage() );}

@Override

public int getItemCount() {

return Array.size();}

static class MyViewHolder extends RecyclerView.ViewHolder implements View.OnClickListener{

private TextView tv, td;

private ImageView ti;

private Context context;

private List<link> Array;

public MyViewHolder(@NonNull View itemView, Context context, List<link> array) {

super(itemView);

this.context = context;

Array = array;

tv = itemView.findViewById(R.id.TVTitle);

td = itemView.findViewById(R.id.Time);

ti = itemView.findViewById(R.id.imageView);

itemView.setOnClickListener(this);}

public void setData(String title, String date, String image){

tv.setText(title);

td.setText(date);

if (Integer.parseInt(image) == 1){ ti.setImageResource(R.drawable.warming);}

else if(Integer.parseInt(image) == 2){ ti.setImageResource(R.drawable.sport); }

else if(Integer.parseInt(image) == 3){ ti.setImageResource(R.drawable.party); }

else{ti.setImageResource(R.drawable.defeult); }}

@Override

public void onClick(View v) {

Intent i = new Intent(context, Edit.class);

i.putExtra(Constant.LIST\_ITEM\_INTENT, Array.get(getAdapterPosition()));

i.putExtra(Constant.EDIT\_STATE, false);

context.startActivity(i); }}

public void updAdapter(List<link> newList){

Array.clear();

Array.addAll(newList);

notifyDataSetChanged();}

public void remove(int pos, DBManager DBM){

DBM.delete(Array.get(pos).getId());

Array.remove(pos);

notifyItemRangeChanged(0, Array.size());

notifyItemRemoved(pos); }}

Reminder.java

package com.example.wtf.adapter;

import android.content.BroadcastReceiver;

import android.content.Context;

import android.content.Intent;

import androidx.core.app.NotificationCompat;

import androidx.core.app.NotificationManagerCompat;

import com.example.wtf.R;

public class Reminder extends BroadcastReceiver {

public static List<Reminder> r = new List<Reminder>(){};

public static int arrayid = 0;

public String title = "У вас важное уведомление!";

public String desc = "Зайдите в приложение";

public int id = 1;

public int image = 1;

public long date = 0;

public void create(String t, String d, int im, int ida, long date){

for (int j = 0; j<r.length; j++){

if (r[j].title == "У вас важное уведомление!"){

r[j].title = t;

r[j].desc = d;

r[j].id = ida;

r[j].image = im;

r[j].date = date;

arrayid = j;

break; }}

sort();}

@Override

public void onReceive(Context context, Intent intent) {

NotificationCompat.Builder builder = new NotificationCompat.Builder(context, "1"+ r[first()].id)

.setSmallIcon(R.drawable.warming)

.setContentTitle(r[first()].title)

.setContentText(r[first()].desc)

.setAutoCancel(true)

.setPriority(NotificationCompat.PRIORITY\_DEFAULT);

if (r[first()].image == 1){ builder.setSmallIcon(R.drawable.warming); }

else if(r[first()].image == 2){ builder.setSmallIcon(R.drawable.sport); }

else if(r[first()].image == 3){ builder.setSmallIcon(R.drawable.party); }

else{ builder.setSmallIcon(R.drawable.defeult); }

r[first()] = new Reminder();

NotificationManagerCompat notificationManagerCompat = NotificationManagerCompat.from(context);

notificationManagerCompat.notify(r[first()].id, builder.build());}

public int first(){

sort();

for (int j = 0; j<r.length; j++) {

if (r[j].title != "У вас важное уведомление!") {return j; }}

return 0; }

public void sort(){

for (int i = 0; i < r.length; i++) {

long min = r[i].date;

int min\_i = i;

for (int j = i+1; j < r.length; j++) {

if (r[j].date < min) {

min = r[j].date;

min\_i = j; }}

if (i != min\_i) {

Reminder tmp = r[i];

r[i] = r[min\_i];

r[min\_i] = tmp; }}}}

Constant.java

package com.example.wtf.db;

public class Constant {

public static final String TABLE\_NAME = "task\_tabele";

public static final String LIST\_ITEM\_INTENT = "list\_item\_intent";

public static final String EDIT\_STATE = "edit\_state";

public static final String ID = "id";

public static final String TITLE = "title";

public static final String DISC = "disc";

public static final String DATE = "date";

public static final String IMAGE = "image";

public static final String NOTIFI = "notify";

public static final String DB\_NAME = "my\_db.db";

public static final int DB\_VERSION = 1;

public static final String TABLE\_STRUCTURE = "CREATE TABLE IF NOT EXISTS " +

TABLE\_NAME + " (" + ID + " INTEGER PRIMARY KEY," + TITLE + " TEXT, "+

DISC + " TEXT, "+ DATE + " TEXT, " + IMAGE + " TEXT," + NOTIFI+ " INTEGER)";

public static final String DROP\_TABLE = "DROP TABLE IF EXISTS " + TABLE\_NAME;}

DBHelper.java

package com.example.wtf.db;

import android.content.Context;

import android.database.sqlite.SQLiteDatabase;

import android.database.sqlite.SQLiteOpenHelper;

import androidx.annotation.Nullable;

public class DBHelper extends SQLiteOpenHelper {

public DBHelper(@Nullable Context context) {

super(context, Constant.DB\_NAME, null,Constant.DB\_VERSION); }

@Override

public void onCreate(SQLiteDatabase db) { db.execSQL(Constant.TABLE\_STRUCTURE); }

@Override

public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {

db.execSQL(Constant.TABLE\_STRUCTURE);

onCreate(db); }}

DBManager.java

package com.example.wtf.db;

import android.content.ContentValues;

import android.content.Context;

import android.database.Cursor;

import android.database.sqlite.SQLiteDatabase;

import com.example.wtf.adapter.link;

import java.util.ArrayList;

import java.util.List;

public class DBManager {

private Context context;

private DBHelper DBHelper;

private SQLiteDatabase db;

public DBManager(Context context){

this.context = context;

DBHelper = new DBHelper(context); }

public void openDB(){db = DBHelper.getReadableDatabase();}

public void delete(int id){

String select = Constant.ID + "=" + id;

db.delete(Constant.TABLE\_NAME, select, null); }

public void update(String title, String disc, String date, String img, int isNotifi,int id){

String select = Constant.ID + "=" + id;

ContentValues cv = new ContentValues();

cv.put(Constant.TITLE, title);

cv.put(Constant.DISC, disc);

cv.put(Constant.DATE, date);

cv.put(Constant.IMAGE, img);

cv.put(String.valueOf(Constant.NOTIFI), isNotifi);

db.update(Constant.TABLE\_NAME, cv,select, null); }

public void insertToDB(String title, String disc, String date, String img, int isNotifi){

ContentValues cv = new ContentValues();

cv.put(Constant.TITLE, title);

cv.put(Constant.DISC, disc);

cv.put(Constant.DATE, date);

cv.put(Constant.IMAGE, img);

cv.put(String.valueOf(Constant.NOTIFI), isNotifi);

db.insert(Constant.TABLE\_NAME, null , cv); }

public List<link> getFromDb() {

List<link> tempList = new ArrayList<>();

Cursor cursor = db.query(Constant.TABLE\_NAME, null , null , null, null, null,null);

while (cursor.moveToNext()){

link l = new link();

String title = cursor.getString(cursor.getColumnIndex(Constant.TITLE));

String desc = cursor.getString(cursor.getColumnIndex(Constant.DISC));

String date = cursor.getString(cursor.getColumnIndex(Constant.DATE));

String img = cursor.getString(cursor.getColumnIndex(Constant.IMAGE));

int isNotifi = cursor.getInt(cursor.getColumnIndex(String.valueOf(Constant.NOTIFI)));

int id = cursor.getInt(cursor.getColumnIndex(Constant.ID));

l.setTitle(title);

l.setDesc(desc);

l.setDate(date);

l.setImage(img);

l.setNotifi(isNotifi);

l.setId(id);

tempList.add(l); }

cursor.close();

return tempList; }

public void closeDB(){DBHelper.close();}}

Edit.java

package com.example.wtf;

import androidx.annotation.RequiresApi;

import androidx.appcompat.app.AppCompatActivity;

import android.app.AlarmManager;

import android.app.DatePickerDialog;

import android.app.NotificationChannel;

import android.app.NotificationManager;

import android.app.PendingIntent;

import android.app.TimePickerDialog;

import android.content.Context;

import android.content.Intent;

import android.os.Build;

import android.os.Bundle;

import android.text.format.DateUtils;

import android.view.View;

import android.widget.ArrayAdapter;

import android.widget.Button;

import android.widget.CheckBox;

import android.widget.DatePicker;

import android.widget.EditText;

import android.widget.Spinner;

import android.widget.TimePicker;

import android.widget.Toast;

import com.example.wtf.adapter.Reminder;

import com.example.wtf.adapter.link;

import com.example.wtf.db.Constant;

import com.example.wtf.db.DBManager;

import java.time.Instant;

import java.util.Calendar;

import java.util.Locale;

public class Edit extends AppCompatActivity {

private EditText edName, edDisk;

private DBManager dbm;

private CheckBox edNotifi;

private Button edDate;

private Spinner spinner;

private View TimeLayout;

private boolean isEditState = true;

private link item;

public long itemms = 0;

static int isNotifi = 0;

long now = 0;

Calendar dateAndTime=Calendar.getInstance();

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_edit);

init();

getmyIntent();}

private void init(){

TimeLayout = findViewById(R.id.TimeLayout);

TimeLayout.setVisibility(View.INVISIBLE);

dbm = new DBManager(this);

edDisk = findViewById(R.id.edDick);

edName = findViewById(R.id.edName);

spinner = findViewById(R.id.SpinnerImage);

edNotifi = findViewById(R.id.NotifiCheckBox);

edDate = findViewById(R.id.DataText);

ArrayAdapter<?> adapter =

ArrayAdapter.createFromResource(this, R.array.image\_list,

android.R.layout.simple\_spinner\_item);

adapter.setDropDownViewResource(android.R.layout.simple\_spinner\_dropdown\_item);

spinner.setAdapter(adapter);

if (edNotifi.isChecked()){

isNotifi = 1;

TimeLayout.setVisibility(View.VISIBLE);}

else{

isNotifi = 0;

TimeLayout.setVisibility(View.INVISIBLE);}}

private void getmyIntent(){

Intent i = getIntent();

if (i!= null){

item = (link)i.getSerializableExtra(Constant.LIST\_ITEM\_INTENT);

isEditState = i.getBooleanExtra(Constant.EDIT\_STATE, true);

if(!isEditState){

edName.setText(item.getTitle());

edDisk.setText(item.getDesc());

edDate.setText(item.getDate());

edNotifi.setChecked(item.getNotifi());

spinner.setSelection(Integer.parseInt(item.getImage()));

if (edNotifi.isChecked()){

isNotifi = 1;

TimeLayout.setVisibility(View.VISIBLE); }

else{

isNotifi = 0;

TimeLayout.setVisibility(View.INVISIBLE); }}} }

protected void onResume() {

super.onResume();

dbm.openDB();}

@RequiresApi(api = Build.VERSION\_CODES.O)

public void onClickSave(View v){

if (edNotifi.isChecked()){

isNotifi = 1;

TimeLayout.setVisibility(View.VISIBLE); }

else{

isNotifi = 0;

TimeLayout.setVisibility(View.INVISIBLE); }

if((edName.getText().toString().equals("")) || (edDate.getText().toString().toUpperCase(Locale.ROOT).equals("ВРЕМЯ") && (isNotifi != 0))){

Toast.makeText(this, R.string.text\_empty, Toast.LENGTH\_SHORT).show();}

else {

if (isNotifi == 1){

int canal = (int) Math.ceil((Math.random()\*1000));

SendMess((itemms / 1000) - Instant.now().getEpochSecond(),edName.getText().toString(),edDisk.getText().toString(), spinner.getSelectedItemPosition(),canal,itemms);}

if(isEditState){

dbm.insertToDB(edName.getText().toString(),edDisk.getText().toString(), edDate.getText().toString(), String.valueOf(spinner.getSelectedItemPosition()), isNotifi);

Toast.makeText(this, R.string.text\_save, Toast.LENGTH\_SHORT).show();

} else{

dbm.update(edName.getText().toString(),edDisk.getText().toString(), edDate.getText().toString(), String.valueOf(spinner.getSelectedItemPosition()), isNotifi, item.getId());

Toast.makeText(this, R.string.text\_update, Toast.LENGTH\_SHORT).show();}

finish();

dbm.closeDB();}}

public void setTime(View v) {

new DatePickerDialog(Edit.this, d,

dateAndTime.get(Calendar.YEAR),

dateAndTime.get(Calendar.MONTH),

dateAndTime.get(Calendar.DAY\_OF\_MONTH))

.show();

new TimePickerDialog(Edit.this, t,

dateAndTime.get(Calendar.HOUR\_OF\_DAY),

dateAndTime.get(Calendar.MINUTE), true)

.show();}

private void setInitialDateTime() {

edDate.setText(DateUtils.formatDateTime(this,

dateAndTime.getTimeInMillis(),

DateUtils.FORMAT\_SHOW\_DATE | DateUtils.FORMAT\_SHOW\_YEAR

| DateUtils.FORMAT\_SHOW\_TIME));

itemms = dateAndTime.getTimeInMillis();}

TimePickerDialog.OnTimeSetListener t = new TimePickerDialog.OnTimeSetListener() {

public void onTimeSet(TimePicker view, int hourOfDay, int minute) {

dateAndTime.set(Calendar.HOUR\_OF\_DAY, hourOfDay);

dateAndTime.set(Calendar.MINUTE, minute);

setInitialDateTime();}};

DatePickerDialog.OnDateSetListener d = new DatePickerDialog.OnDateSetListener() {

public void onDateSet(DatePicker view, int year, int monthOfYear, int dayOfMonth) {

dateAndTime.set(Calendar.YEAR, year);

dateAndTime.set(Calendar.MONTH, monthOfYear);

dateAndTime.set(Calendar.DAY\_OF\_MONTH, dayOfMonth);

setInitialDateTime(); }};

private void SendMess(long inter, String title, String desc, int img, int id, long date){

createNotificatinChannel(id);

Reminder r = new Reminder();

r.create(title,desc,img,id,date);

Intent intent = new Intent(this, Reminder.class);

PendingIntent pendingIntent = PendingIntent.getBroadcast(this,id,intent,id);

now = System.currentTimeMillis();

AlarmManager alarmReceiver = (AlarmManager) getSystemService(ALARM\_SERVICE);

alarmReceiver.set(AlarmManager.RTC\_WAKEUP, now + (inter \* 1000), pendingIntent ); }

private void createNotificatinChannel(int id){

NotificationManager notificationManager =

(NotificationManager) getSystemService(Context.NOTIFICATION\_SERVICE);

if (android.os.Build.VERSION.SDK\_INT >= android.os.Build.VERSION\_CODES.O) {

NotificationChannel channel = new NotificationChannel("1"+id, "1",

NotificationManager.IMPORTANCE\_HIGH);

channel.setDescription("My channel description");

channel.enableLights(true);

channel.enableVibration(true);

notificationManager.createNotificationChannel(channel); }}

public void CheckBoxNotify(View v){

if (edNotifi.isChecked()){

isNotifi = 1;

TimeLayout.setVisibility(View.VISIBLE); }

else{

isNotifi = 0;

TimeLayout.setVisibility(View.INVISIBLE); }}}

Loading.java

package com.example.wtf;

import androidx.appcompat.app.AppCompatActivity;

import android.app.NotificationChannel;

import android.app.NotificationManager;

import android.content.Context;

import android.content.Intent;

import android.os.Bundle;

public class Loading extends AppCompatActivity {

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

createNotificatinChannel();

Intent i = new Intent(this, MainActivity.class);

startActivity(i); }

private void createNotificatinChannel(){

NotificationManager notificationManager =

(NotificationManager) getSystemService(Context.NOTIFICATION\_SERVICE);

if (android.os.Build.VERSION.SDK\_INT >= android.os.Build.VERSION\_CODES.O) {

NotificationChannel channel = new NotificationChannel("1", "My channel",

NotificationManager.IMPORTANCE\_HIGH);

channel.setDescription("My channel description");

channel.enableLights(true);

channel.enableVibration(true);

notificationManager.createNotificationChannel(channel); }}

MainActivity.java

package com.example.wtf;

import androidx.annotation.NonNull;

import androidx.annotation.RequiresApi;

import androidx.appcompat.app.AppCompatActivity;

import androidx.recyclerview.widget.ItemTouchHelper;

import androidx.recyclerview.widget.LinearLayoutManager;

import androidx.recyclerview.widget.RecyclerView;

import android.content.Intent;

import android.os.Build;

import android.os.Bundle;

import android.view.View;

import android.widget.Button;

import android.widget.CheckBox;

import android.widget.EditText;

import android.widget.Spinner;

import android.widget.TextView;

import com.example.wtf.adapter.MainAdapter;

import com.example.wtf.db.DBManager;

import java.text.DateFormat;

import java.text.SimpleDateFormat;

import java.util.Calendar;

import java.util.Date;

import java.util.Locale;

public class MainActivity extends AppCompatActivity {

private DBManager dbm;

private TextView date\_mounth, date\_year, date\_day, date\_name;

private RecyclerView rcView;

private MainAdapter mainAdapter;

@RequiresApi(api = Build.VERSION\_CODES.O)

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

init();}

@RequiresApi(api = Build.VERSION\_CODES.O)

private void init() {

dbm = new DBManager(this);

date\_mounth = findViewById(R.id.dateMounth);

date\_year = findViewById(R.id.dateYear);

date\_day = findViewById(R.id.dateDay);

date\_name = findViewById(R.id.DayName);

rcView = findViewById(R.id.rcView);

mainAdapter = new MainAdapter(this);

rcView.setLayoutManager(new LinearLayoutManager(this));

getItemTouchHelper().attachToRecyclerView(rcView);

rcView.setAdapter(mainAdapter);

Date currentDate = new Date();

DateFormat dateDay = new SimpleDateFormat("dd", Locale.getDefault());

Calendar calendar = Calendar.getInstance();

String dateM = calendar.getDisplayName(Calendar.MONTH,

Calendar.LONG\_FORMAT, new Locale("ru"));

DateFormat dateY = new SimpleDateFormat("yyyy", Locale.getDefault());

DateFormat dateName = new SimpleDateFormat("EEEE", Locale.getDefault());

String date\_d = dateDay.format(currentDate);

String date\_nameStr = dateName.format(currentDate);

String date\_y = dateY.format(currentDate);

date\_day.setText(date\_d);

date\_year.setText(date\_y);

String mounth\_name = dateM.toLowerCase(Locale.ROOT);

date\_mounth.setText(mounth\_name);

date\_name.setText(date\_nameStr); }

@Override

protected void onResume() {

super.onResume();

dbm.openDB();

mainAdapter.updAdapter(dbm.getFromDb()); }

public void onClickAdd(View view) {

Intent i = new Intent(MainActivity.this, Edit.class);

startActivity(i); }

protected void onDestroy() {

super.onDestroy();

dbm.closeDB();}

private ItemTouchHelper getItemTouchHelper() {

return new ItemTouchHelper(new ItemTouchHelper.SimpleCallback(0, ItemTouchHelper.LEFT | ItemTouchHelper.RIGHT) {

@Override

public boolean onMove(@NonNull RecyclerView recyclerView, @NonNull RecyclerView.ViewHolder viewHolder, @NonNull RecyclerView.ViewHolder target) {

return false; }

@Override

public void onSwiped(@NonNull RecyclerView.ViewHolder viewHolder, int direction) {

mainAdapter.remove(viewHolder.getAdapterPosition(), dbm); }})}}

Приложение Б. Руководство пользователя

Руководство пользователя мобильного приложения.

1. Введение.

Область применения приложения: программа используется потребителем любой возрастной категории, для записи нужных им заметок и задач.

Описание возможностей: программа может создавать заметки и задачи, добавлять к ним уведомления, записывать им описание и выбирать дизайн. Любую запись можно удалить или отредактировать.

Уровень подготовки пользователя: низкий.

1. Назначение и условия применения.

Виды деятельности, функции, для автоматизации которых предназначено данное средство автоматизации: создание необходимых записей для запоминания и напоминания.

Требования к применению: наличие мобильного устройства с операционной системой Android с версией выше 5.0 и объёмом свободной постоянной памяти больше 10Мб.

1. Подготовка к работе.

Порядок загрузки: установить файл apk на телефон и по нажатию на него произвести установку программы.

Порядок проверки работоспособности: запуск приложения и создание тестовой заметки с выбором наличия уведомления.

1. Описание операций.

Создание заметки: требуется нажать на кнопку добавить на главном экране приложения, после чего откроется окно редактирования, в котором требуется ввести необходимые данные.

Сохранение заметки: в окне редактирования, после ввода данных нажать на кнопку сохранения.

Редактирование заметки: нажать на существующую заметку, после чего откроется окно редактирования, на котором можно внести необходимые изменение, после чего требуется нажать на кнопку сохранения.

Удаление заметки: на главном экране осуществить свайп по существующей заметке в левую или правую сторону.

1. Аварийные ситуации.

При возникновении аварийной ситуации требуется перезапустить приложение, если приложение продолжает функционировать аварийно требуется переустановить программу.